ÖZYEĞİN ÜNİVERSİTESİ

EE321 - Microprocessors

Laboratory 1

Spring 2023

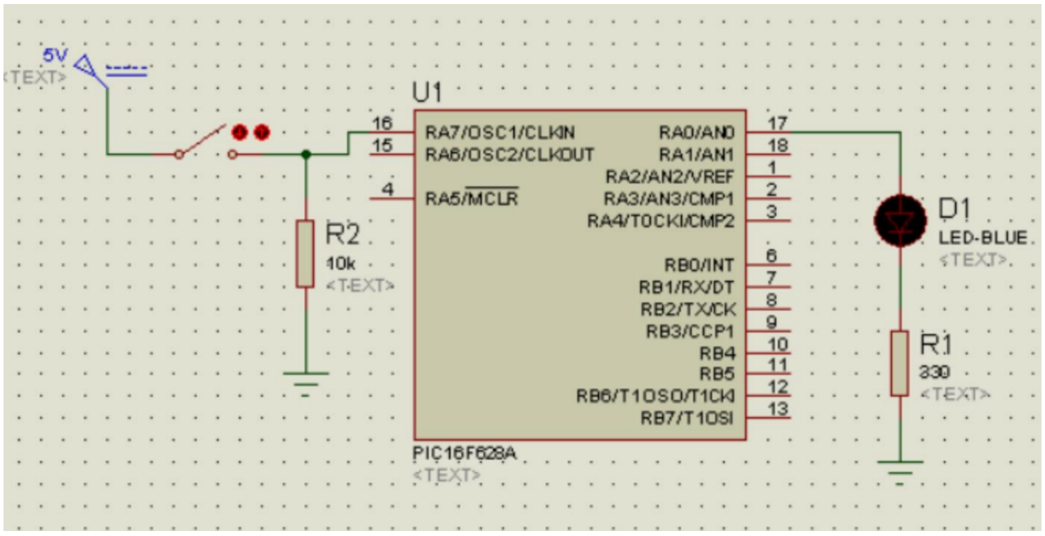
1 Introduction

This lab introduces the concepts and general practices for programming and simulating PIC microcontrollers using the mikroC PRO IDE (Integrated Development Environment) and Proteus. In this lab, we will learn the basic digital input and output capability of an MCU. Digital input pins are the base component for reading an external digital input. Digital devices such as push buttons are connected to these pins. Digital output pins are the base component for sending a digital signal to an external component. Digital devices such as LEDs are connected to these pins.

1.1 Lab Tasks

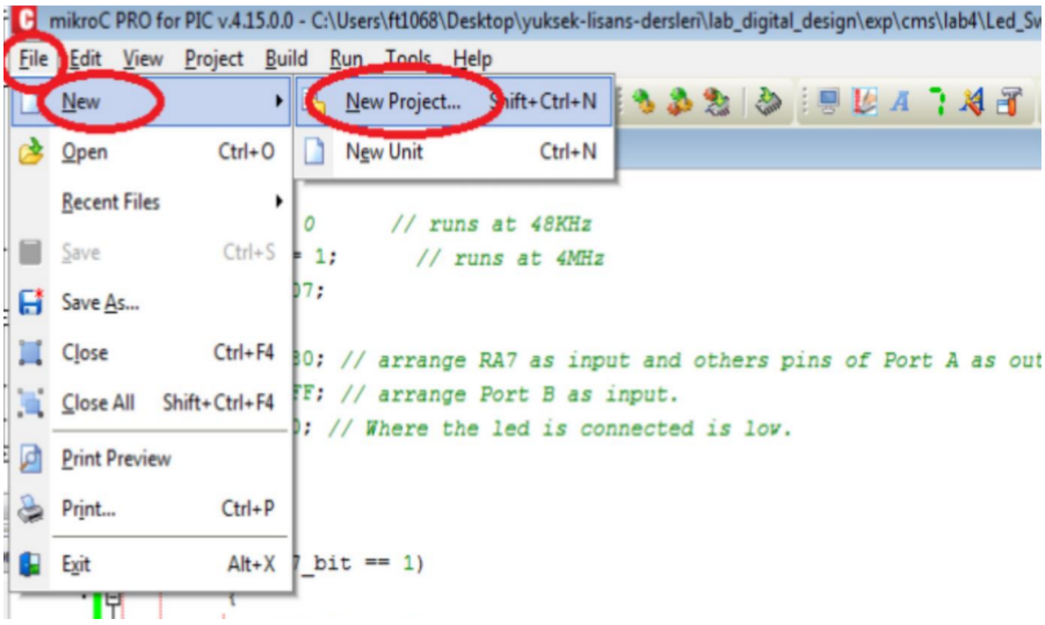
1.1.1 Task 1:

In Part 1 of this lab, you will learn how to make your first schematic design for simulation using Proteus and practice with a basic design. After completing Part 1, you will learn how to compose a program for PIC MCU using the MikroC PRO IDE tool and simulate it on Proteus.



Setup and open mikroC (microC_PRO_for_PIC).

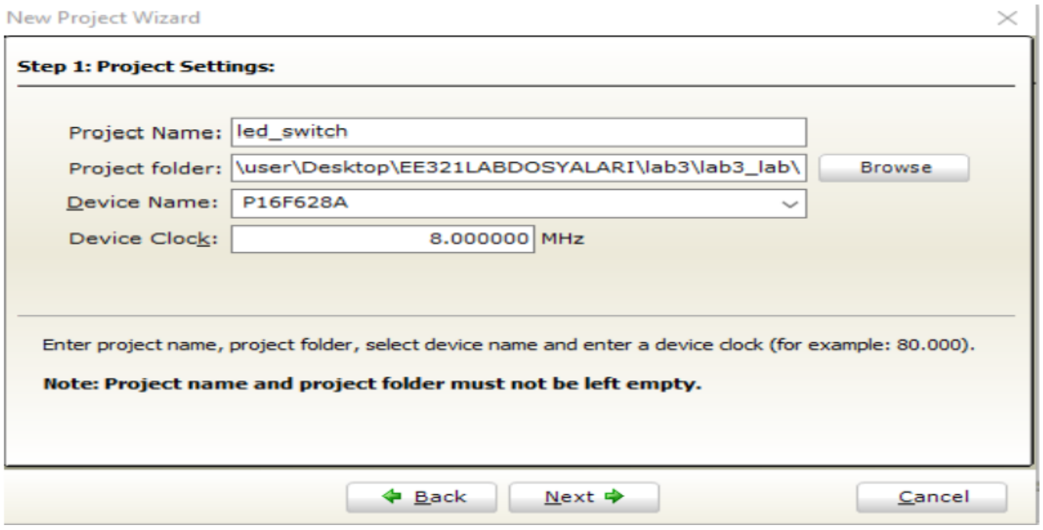
1. Create a new project.


mikroC PRO for PIC v.4.15.0.0 - C:\Users\ft1068\Desktop\yuksek-lisans-dersleri\lab_digital_design\exp\cms\lab4\Led_Sv
File Edit View Project Build Run Tools Help
New
New Project... Shift+Ctrl+N
New Unit Ctrl+N
Open Ctrl+O
Recent Files
Save Ctrl+S
Save As...
Close Ctrl+F4
Close All Shift+Ctrl+F4
Print Preview
Print... Ctrl+P
Exit Alt+X

```
0       // runs at 48KHz
 1;       // runs at 4MHz
07;

30; // arrange RA7 as input and others pins of Port A as out
FF; // arrange Port B as input.
0; // Where the led is connected is low.

7_bit == 1)
```

2. Then click next then select the device name as P16F628A. Also, set the device clock to 8 MHz.

3. Select the project location where you place your project files, and give your design a name: “led_switch”.


New Project Wizard

Step 1: Project Settings:

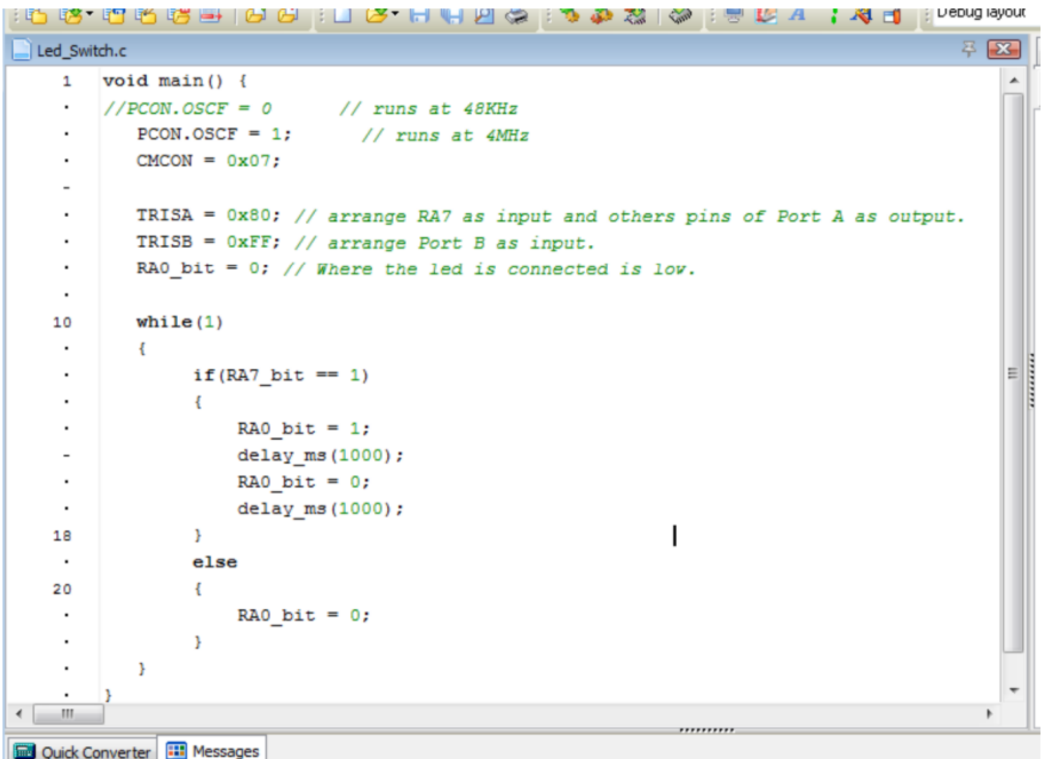
Project Name:	led_switch
Project folder:	\user\Desktop\EE321LABDOSYALARI\lab3\lab3_lab\
Device Name:	P16F628A
Device Clock:	8.000000 MHz

Enter project name, project folder, select device name and enter a device clock (for example: 80.000).

Note: Project name and project folder must not be left empty.

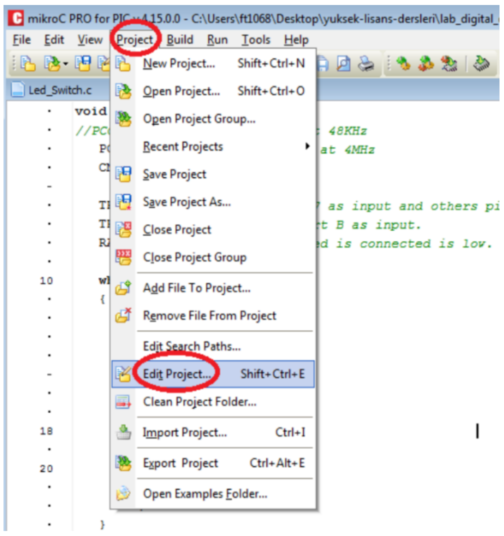
4. Then click ”Next” three times and then ”Finish”. In the end, your window should be like the following.

5. Write the following code. You can find the code on the LMS.

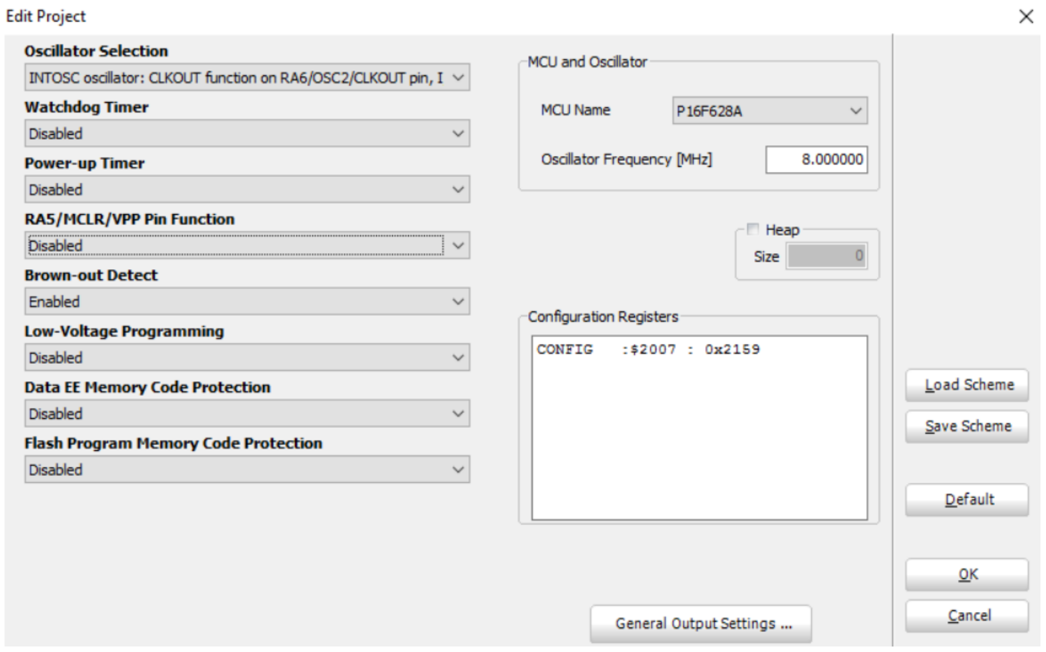


```
void main() {
//PCON.OSCF = 0       // runs at 48KHz
   PCON.OSCF = 1;       // runs at 4MHz
   CMCON = 0x07;

   TRISA = 0x80; // arrange RA7 as input and others pins of Port A as output.
   TRISB = 0xFF; // arrange Port B as input.
   RA0_bit = 0; // Where the led is connected is low.

   while(1)
   {
        if(RA7_bit == 1)
        {
            RA0_bit = 1;
            delay_ms(1000);
            RA0_bit = 0;
            delay_ms(1000);
        }
        else
        {
            RA0_bit = 0;
        }
   }
}
```

6. Then click Project → Edit Project



7. A new window should open. Configure it like the below. And click OK.



Edit Project

Oscillator Selection	INTOSC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, I
Watchdog Timer	Disabled
Power-up Timer	Disabled
RA5/MCLR/VPP Pin Function	Disabled
Brown-out Detect	Enabled
Low-Voltage Programming	Disabled
Data EE Memory Code Protection	Disabled
Flash Program Memory Code Protection	Disabled

MCU and Oscillator

MCU Name: P16F628A

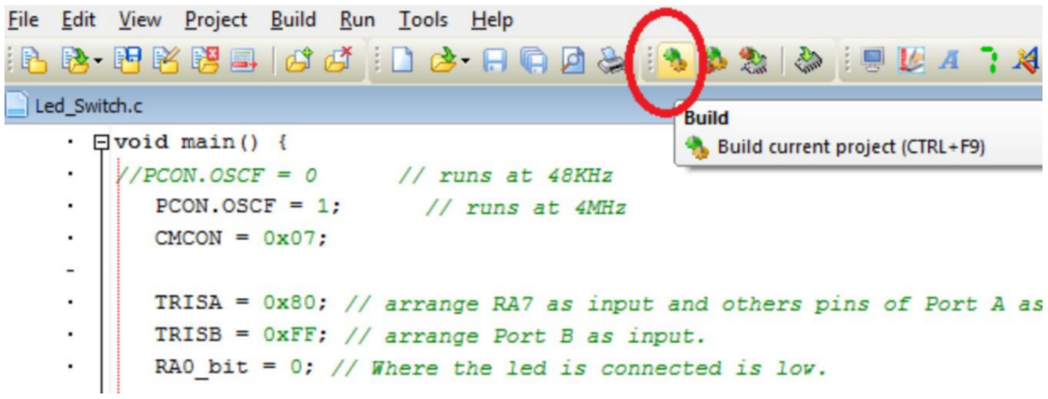
Oscillator Frequency [MHz]: 8.000000

Heap Size: 0

Configuration Registers

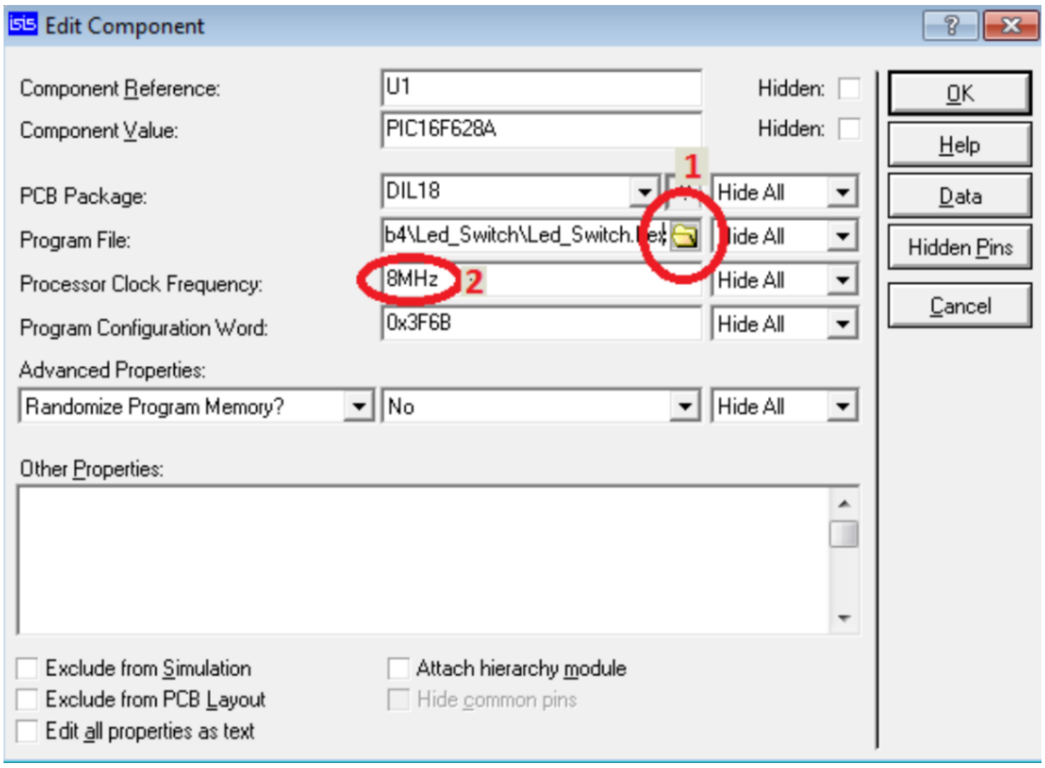
```
CONFIG    :$2007 : 0x2159
```

8. Then click the icon circled on the picture below to generate the .hex file. This hex file will be sent to PIC later.



```
void main() {
//PCON.OSCF = 0       // runs at 48KHz
   PCON.OSCF = 1;       // runs at 4MHz
   CMCON = 0x07;

   TRISA = 0x80; // arrange RA7 as input and others pins of Port A as
   TRISB = 0xFF; // arrange Port B as input.
   RA0_bit = 0; // Where the led is connected is low.
```

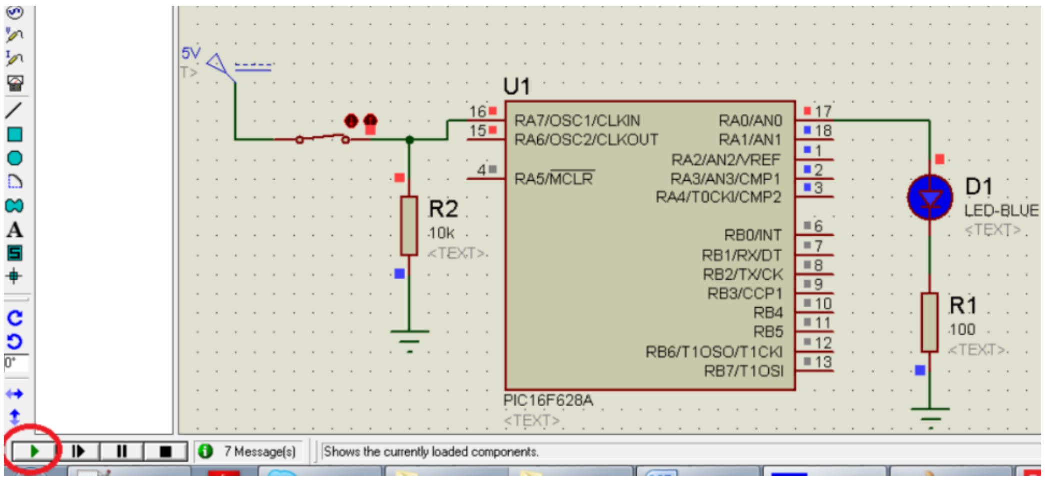
9. Now, open Led_Switch DSN file. You will see the Proteus page. And double click on PIC schematic symbol. A new window will appear. Click the folder icon to select the .hex file you created. You can find this .hex file in your mikroC project location. And click OK.



Edit Component

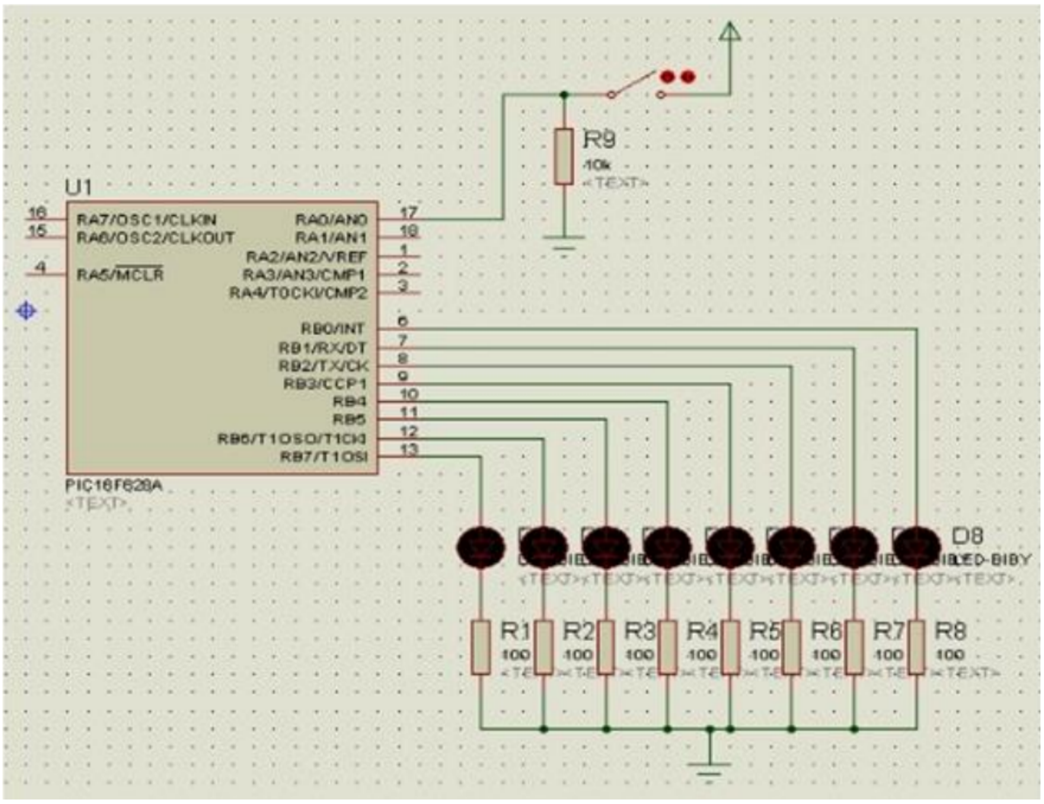
Component Reference:	U1	Hidden:	
Component Value:	PIC16F628A	Hidden:	
PCB Package:	DIL18	Hide All	
Program File:	b4\Led_Switch\Led_Switch.hex	Hide All	
Processor Clock Frequency:	8MHz	Hide All	
Program Configuration Word:	0x3F6B	Hide All	
Randomize Program Memory?	No	Hide All	

10. To start the simulation, you should click the play icon circled below.



1.1.2 Task 2:

1. Open a new Proteus schematic and draw the following PIC circuit.



2. Create a new MikroC project and name it "Rotating_Dot". Apply the same steps in part 1 on the pdf.

3. There should be a demo video showing the functionality in the lab 1 folder. The functionality is as follows:

   (a) There should be only one LED active at any given time
   (b) Active LED (a.k.a. the dot) should move to the next LED every second
   (c) When the dot arrives at one edge, it should jump back to the other edge and continue moving.
   (d) State of the switch should change the direction of the dot

4. Write the code and simulate it in Proteus. Submit your files as described in section 1.1.4.

1.1.3 Taks 3:

1. Create a new MikroC project and name it "Knight_Rider". Apply the same steps in part 1 on the pdf.

2. There should be a demo video showing the functionality in the lab 1 folder. The functionality is as follows:

   (a) There should be only one LED active at any given time

(b) Active LED (a.k.a. the dot) should move to the next LED every second

(c) When the dot arrives at one edge, it should start moving in the opposite direction.

(d) When the switch is closed, the dot should stop moving. The dot should continue moving when it is open.

3. Write the code and simulate it in Proteus. Submit your files as described in section 1.1.4.

1.1.4 What to submit

You should submit five files in total.

1. Proteus Design File
2. RotatingDot C File
3. RotatingDot HEX File
4. KnightRider C File
5. KnightRider HEX File

The name of the compressed ZIP or RAR file must be your student number: S012345.zip or S012345.rar

Code sharing is strictly forbidden. Your codes will be correlated and graded accordingly.

1.2 Background

1.2.1 What is microcontroller?

Microcontrollers (a.k.a. Micro Controller Units, i.e., MCUs) are used in automatically controlled products and devices include automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys, and other embedded systems. MCUs offer implementations with reduced size and cost compared to implementations using a separate microprocessor (uP), memory, and I/O devices. MCUs are usually mixed-signal, that is, they integrate analog components needed to control non-digital electronic systems. An MCU has a processor along with internal memory and I/O components (Fig. 1b). An uP is the heart of a computer system. It is just a processor; memory and I/O components have to be connected externally (Fig. 1a). uPs usually run at much faster clock frequencies than MCUs and have complicated memory subsystem and features, all geared towards running programs fast. uPs usually consume a lot more power than MCUs and are not a good choice for compact devices and embedded systems. There are several MCU vendors like TI, Microchip, Maxim-IC, and so on. We at Ozyegin (and also in Turkey overall) use PIC MCUs, which are manufactured by a company called Microchip.

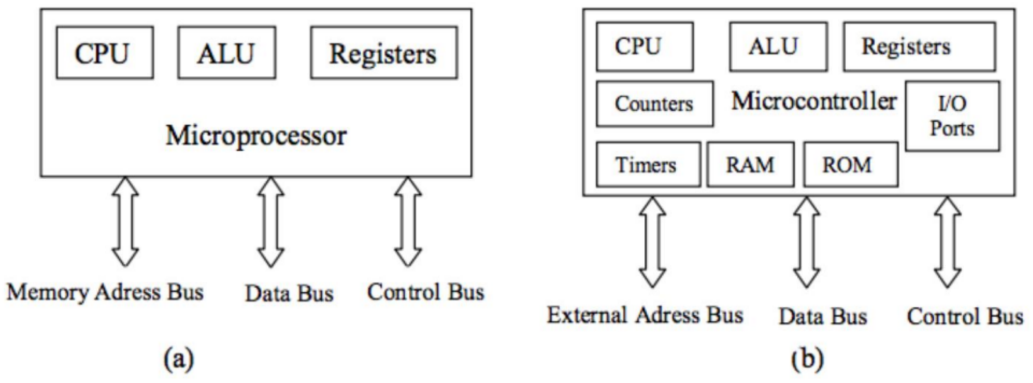


Figure 1. (a) Structure of a microprocessor. (b) Structure of a microcontroller.

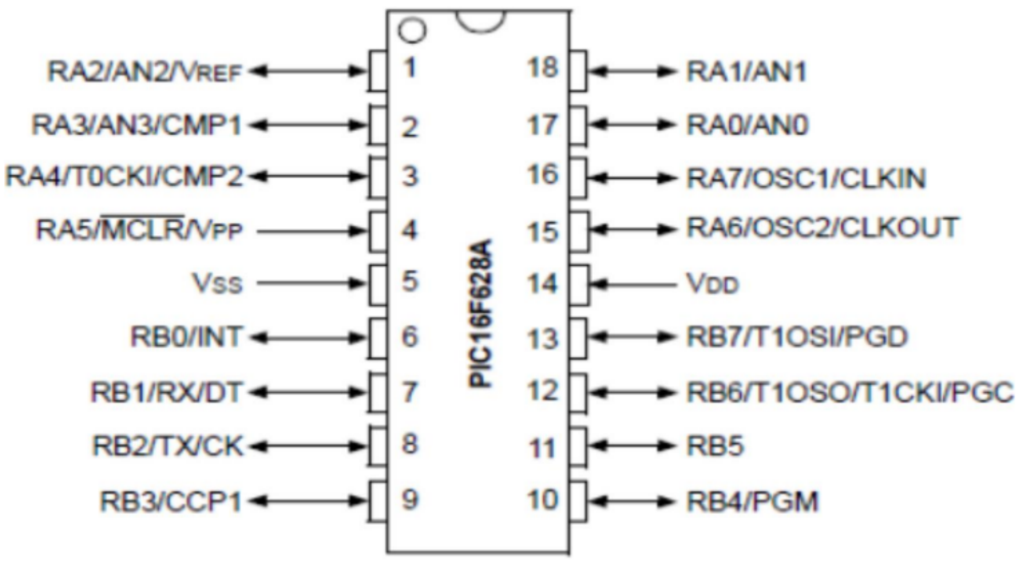
1.2.2 PIC Microcontrollers

PICs are popular with industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.

There are several PIC families. However, for this week, we will use PIC16F628A. Above is a diagram showing the pin-outs of the PIC16F628A. We will go through some pins, explaining what they are used for. Most pins of the PIC handle more than one function. For example, pin 1 has three different functions, namely, RA2, AN2, and VREF.

VSS and VDD: These are the power supply pins. VDD is the positive supply, and VSS is the negative supply or 0V. The maximum supply voltage that you can use is 6V, and the minimum is 2V. RA0 to RA7 and RB0 to RB7: These are bidirectional pins. That is, they can be configured as an input or an output. The number following RA and RB are the bit numbers (0 to 7). Hence, we have 16-bit directional pins where each bit can be configured as digital input or output.
RX and TX: You can use these pins for serial communication to other devices, provided you configure the PIC to use this function.

AN0 to AN1: If you need an analog comparator, you can use these pins as analog I/O.
MCLR: You can use this pin to reset the PIC, provided you configure the PIC to use this function.
OSC1 and OSC2: If you want to create your own clock using a crystal, you can use these pins for that, provided you configure the PIC to use this function.



1.2.3 PIC programming

Creating New File:

File → New Project → Next → Fill the Project Name → Browse Project Folder → Select Device → Select Device Clock → No operation is needed for steps 2 3 4. Please skip these steps with the ”Next Button.”
Device Selection: There are many MCU types in the Microchip PIC family. Each PIC family has a different instruction set. When we choose the type of PIC from this list, the compiler uses the appropriate instruction set. In this lab, we will use the PIC16F628A microcontroller.
Device Clock: MCUs need a repetitive square signal to operate. Oscillators can produce this repetitive signal for MCUs. There are two types of oscillators that can be used: external and internal. Both of their frequencies can be adjusted. Oscillator selection affects the stability and reliability of the system's operation. Internal oscillators consist of resistors and capacitors. Although the accuracy of external oscillators is very high, the accuracy of internal oscillators can easily vary due to environmental conditions. One of the most conspicuous examples is temperature. Values of resistors and capacitors change with temperature. Therefore, MCUs that work with internal oscillators do not work properly after a certain temperature. In these conditions, due to the fact that the stability of external oscillators is much higher under environmental fluctuations, working with an external oscillator may be a better choice. On the other hand, internal oscillator use reduces the size and cost of the circuit.

Internal Oscillator Configurations for PIC16F628A: (After creating a new project) Project → Edit Project → Oscillator Selection: INTOSC oscillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN → Watchdog Timer: disabled → RA5/MCLR/VPP Pin Function: Disabled → Brown-out Detect: Disabled

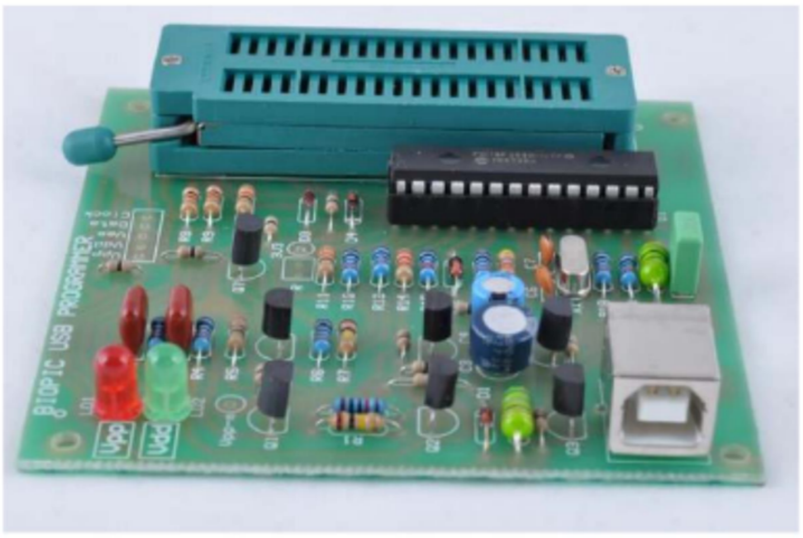


Figure 4. Biopic programmer board.

1.2.4 Programming with mikroC PRO

TRISA (PORT A), TRISB (PORT B), TRISC (PORT C), TRISD (PORT D):
Are registers that hold pins of input and output ports? If a pin of a port is allocated as input, the pin can only receive data; it cannot send any data, and vice versa.

Example:
If we want to assign the following pin configuration to PORTB:

PORTB Pin7 = 1 → Input
PORTB Pin6 = 1 → Input
PORTB Pin5 = 1 → Input
PORTB Pin4 = 1 → Input
PORTB Pin3 = 0 → Output
PORTB Pin2 = 0 → Output
PORTB Pin1 = 0 → Output
PORTB Pin0 = 0 → Output

Then, we can write the pin configuration as binary: 1111 0000, which is equivalent to 240 in decimal. Finally, we assign this value:

TRISB = 240 ;

PORTA & PORTB & PORTC & PORTD are used for sending data to pins of ports or reading data from the ports.

Example:
We want to send the following digital data configuration:

PORTB Pin7 = 0 → 0V
PORTB Pin6 = 0 → 0V
PORTB Pin5 = 0 → 0V
PORTB Pin4 = 0 → 0V
PORTB Pin3 = 0 → 0V

PORTB Pin2 = 1 $\rightarrow$ 5V
PORTB Pin1 = 1 $\rightarrow$ 5V
PORTB Pin0 = 1 $\rightarrow$ 5V

First, all pins are assigned as output:

TRISB = 255; $\rightarrow$ which is the decimal equivalent of $(11111111)_2$
Then, the output configuration is written in binary and converted to decimal:
[Pin7 Pin6 Pin5 Pin4 Pin3 Pin2 Pin1 Pin0] = $(00000111)_2 = 8$

Finally, assign the decimal value to the
PORTB: PORTB = 8;

Sending data to the pins or reading data from the pins can be made collectively as also it can be made separately.